Relatório ALGAV

Sprint 2

Realizado por :

1221352 - Bruno Ribeiro

1230430 - Eduardo Ferreira

1220839 - Vasco Teixeira

1221957 - João Silva

Explicação do código base fornecido para agendar as operações

O código começa por definir os seguintes dados :

```
agenda_staff(d001,20241028,[(720,790,m01),(1080,1140,c01)]).
agenda_staff(d002,20241028,[(850,900,m02),(901,960,m02),(1380,1440,c02)]).
agenda_staff(d003,20241028,[(720,790,m01),(910,980,m02)]).
agenda_staff(d004,20241028,[(850,900,m02),(940,980,c04)]).

timetable(d001,20241028,(480,1200)).
timetable(d002,20241028,(500,1440)).
timetable(d003,20241028,(520,1320)).
timetable(d004,20241028,(620,1020)).

staff(d001,doctor,orthopaedist,[so2,so3,so4]).
staff(d002,doctor,orthopaedist,[so2,so3,so4]).
staff(d003,doctor,orthopaedist,[so2,so3,so4]).
staff(d004,doctor,orthopaedist,[so2,so3,so4]).
```

- **agenda_staff**: Representa a agenda dos médicos para um dia específico, mostrando os horários das cirurgias do médico naquele dia.
- **timetable**: Define o horário de trabalho de cada médico.
- **staff**: Define innformações sobre os médicos, o id do staff, a sua função(doctor,nurse,technician), a sua especialidade e as cirurgias que este pode realizar.

```
surgery(so2,45,60,45).
surgery(so3,45,90,45).
surgery(so4,45,75,45).


surgery_id(so100001,so2).
surgery_id(so100002,so3).
surgery_id(so100003,so4).
surgery_id(so100004,so2).
surgery_id(so100005,so4).
%surgery_id(so100006,so2).
%surgery_id(so100007,so3).
%surgery_id(so100008,so2).
%surgery_id(so100009,so2).
%surgery_id(so100010,so2).
%surgery_id(so100011,so4).
%surgery_id(so100012,so2).
%surgery_id(so100013,so2).

assignment_surgery(so100001,d001).
assignment_surgery(so100002,d002).
assignment_surgery(so100003,d003).
assignment_surgery(so100004,d001).
assignment_surgery(so100004,d002).
assignment_surgery(so100005,d002).
assignment_surgery(so100005,d003).
%assignment_surgery(so100006,d001).
%assignment_surgery(so100007,d003).
%assignment_surgery(so100008,d004).
%ssignment_surgery(so100008,d003).
%assignment_surgery(so100009,d002).
%assignment_surgery(so100009,d004).
%assignment_surgery(so100010,d003).
%assignment_surgery(so100011,d001).
%assignment_surgery(so100012,d001).
%assignment_surgery(so100013,d004).
```

- **surgery**: Define o tipo de cirurgia e os tempos de cada fase da mesma (anestesia, cirurgia e limpeza).
- **surgery_id**: Atribui o id de uma cirurgia a um tipo de cirurgia.
- **assignment_surgery**: Atribui uma cirurgia a um médico específico de acordo com o id da cirurgia e o id do médico.

Estes dados vão ser utilizados nas funções que mostraremos mais abaixo.

```
free_agenda0([],[(0,1440)]).
free_agenda0([(0,Tfin,_)|LT],LT1):-!,free_agenda1([(0,Tfin,_)|LT],LT1).
free_agenda0([(Tin,Tfin,_)|LT],[(0,T1)|LT1]):- T1 is Tin-1,
    free_agenda1([(Tin,Tfin,_)|LT],LT1).

free_agenda1([(_,Tfin,_)],[(T1,1440)]):-Tfin\==1440,!,T1 is Tfin+1.
free_agenda1([(_,_,_)],[]).
free_agenda1([(_,T,_),(T1,Tfin2,_)|LT],LT1):-Tx is T+1,T1==Tx,!,
    free_agenda1([(T1,Tfin2,_)|LT],LT1).
free_agenda1([(_,Tfin1,_),(Tin2,Tfin2,_)|LT],[(T1,T2)|LT1]):-T1 is Tfin1+1,T2 is Tin2-1,
    free_agenda1([(Tin2,Tfin2,_)|LT],LT1).


adapt_timetable(D,Date,LFA,LFA2):-timetable(D,Date,(InTime,FinTime)),treatin(InTime,LFA,LFA1),treatfin(FinTime,LFA1,LFA2).

treatin(InTime,[(In,Fin)|LFA],[(In,Fin)|LFA]):-InTime=<In,!.
treatin(InTime,[(_,Fin)|LFA],LFA1):-InTime>Fin,!,treatin(InTime,LFA,LFA1).
treatin(InTime,[(_,Fin)|LFA],[(InTime,Fin)|LFA]).
treatin(_,[],[]).

treatfin(FinTime,[(In,Fin)|LFA],[(In,Fin)|LFA1]):-FinTime>=Fin,!,treatfin(FinTime,LFA,LFA1).
treatfin(FinTime,[(In,_)|_],[]):-FinTime=<In,!.
treatfin(FinTime,[(In,_)|_],[(In,FinTime)]).
treatfin(_,[],[]).
```

- **free_agenda0/2:** Este predicado inicializa o processo de encontrar horários livres, partindo de uma lista de horários ocupados. O predicado processa cada intervalo de tempo ocupado, e a partir disso, começa a calcular o horário livre. Caso a lista de horários esteja vazio a função assume que o horário disponível para aquele dia é o dia todo, caso contrário chama o predicado free_agenda1/2 para calcular os horários disponíveis a partir do primeiro horário ocupado.

- **free_agenda1/2:** Este predicado lida com o processamento real dos horários ocupados. Ele determina se os horários podem ser combinados ou se há intervalos livres entre as ocupações. Caso exista um intervalo de tempo livre entre o fim de uma operação e o início de outra, ele retorna esse intervalo. Se o tempo de fim de uma cirurgia não for o último horário possível (1440) , ele ajusta o tempo livre para começar um minuto após o fim da cirurgia e termina um minuto antes do início da próxima cirurgia.

- **adapt_timetable/3:** Este predicado ajusta a agenda de um médico, considerando os horários já ocupados e os horários livres. O objetivo é ajustar o horário de um determinado dia para garantir que não haja sobreposição de intervalos entre as agendas dos diferentes médicos e das operações agendadas. As funções treatin/3 e treatfin/3 são chamadas para verificar se o horário de entrada ou de fim de uma operação pode ser encaixado na agenda.

```
schedule_all_surgeries(Room,Day):-
    retractall(agenda_staff1(_,_,_)),
    retractall(agenda_operation_room1(_,_,_)),
    retractall(availability(_,_,_)),
    findall(_,(agenda_staff(D,Day,Agenda),assertz(agenda_staff1(D,Day,Agenda))),_),
    agenda_operation_room(Or,Date,Agenda),assert(agenda_operation_room1(Or,Date,Agenda)),
    findall(_,(agenda_staff1(D,Date,L),free_agenda0(L,LFA),adapt_timetable(D,Date,LFA,LFA2),assertz(availability(D,Date,LFA2))),_),
    findall(OpCode,surgery_id(OpCode,_),LOpCode),

    availability_all_surgeries(LOpCode,Room,Day),!.

availability_all_surgeries([],_,_).
availability_all_surgeries([OpCode|LOpCode],Room,Day):-
    surgery_id(OpCode,OpType),surgery(OpType,_,TSurgery,_),
    availability_operation(OpCode,Room,Day,LPossibilities,LDoctors),
    schedule_first_interval(TSurgery,LPossibilities,(TinS,TfinS)),
    retract(agenda_operation_room1(Room,Day,Agenda)),
    insert_agenda((TinS,TfinS,OpCode),Agenda,Agenda1),
    assertz(agenda_operation_room1(Room,Day,Agenda1)),
    insert_agenda_doctors((TinS,TfinS,OpCode),Day,LDoctors),
    availability_all_surgeries(LOpCode,Room,Day).



availability_operation(OpCode,Room,Day,LPossibilities,LDoctors):-surgery_id(OpCode,OpType),surgery(OpType,_,TSurgery,_),
    findall(Doctor,assignment_surgery(OpCode,Doctor),LDoctors),
    intersect_all_agendas(LDoctors,Day,LA),
    agenda_operation_room1(Room,Day,LAgenda),
    free_agenda0(LAgenda,LFAgRoom),
    intersect_2_agendas(LA,LFAgRoom,LIntAgDoctorsRoom),
    remove_unf_intervals(TSurgery,LIntAgDoctorsRoom,LPossibilities).
```

- **schedule_all_surgeries/2 :** Este predicado organiza o agendamento de todas as cirurgias. Ele é responsável por preencher a agenda dos médicos e a agenda da sala de cirurgia, tendo em conta a disponibilidade de horários. Em seguida, vai buscar os horários dos médicos e salas de cirurgia, e as disponibilidades para as cirurgias. Para cada cirurgia, a função chama availability_all_surgeries/3 para verificar a disponibilidade dos médicos e das salas, e tenta encaixar a cirurgia nos horários livres.

- **availability_all_surgeries/3 :** Esta função verifica a disponibilidade de médicos e salas de operação para agendar uma cirurgia específica. Para cada cirurgia, o predicado verifica as possíveis agendas de médicos e salas através da chamada de availiability_operation/5.

- **availability_operation/5 :** O objetivo deste predicado é determinar quais horários estão disponíveis para agendar uma cirurgia numa sala específica e num dia específico. Duranto o processo temos em conta a agenda dos médicos envolvidos, a sala de cirurgia e o tempo necessário para a cirurgia.

```
insert_agenda((TinS,TfinS,OpCode),[],[(TinS,TfinS,OpCode)]).
insert_agenda((TinS,TfinS,OpCode),[(Tin,Tfin,OpCode1)|LA],[(TinS,TfinS,OpCode),(Tin,Tfin,OpCode1)|LA]):-TfinS<Tin,!.
insert_agenda((TinS,TfinS,OpCode),[(Tin,Tfin,OpCode1)|LA],[(Tin,Tfin,OpCode1)|LA1]):-insert_agenda((TinS,TfinS,OpCode),LA,LA1).

insert_agenda_doctors(_,_,[]).
insert_agenda_doctors((TinS,TfinS,OpCode),Day,[Doctor|LDoctors]):-
    retract(agenda_staff1(Doctor,Day,Agenda)),
    insert_agenda((TinS,TfinS,OpCode),Agenda,Agenda1),
    assert(agenda_staff1(Doctor,Day,Agenda1)),
    insert_agenda_doctors((TinS,TfinS,OpCode),Day,LDoctors).
```

- **insert_agenda/3:** Esta função insere uma cirurgia na agenda (quer dos médicos quer das salas). Se a lista de agendas estiver vazia, ela simplesmente adiciona o novo intervalo de tempo da cirurgia. Caso contrário, ela percorre a lista de agendas já existentes e tenta encaixar o novo intervalo de tempo no horário disponível, comparando o fim de uma operação e o início de outra.

- **insert_agenda_doctors/3:** Essa função insere o horário da cirurgia nas agendas dos médicos que foram selecionados. Para cada médico, a função chama insert_agenda/3 para adicionar o intervalo de tempo da cirurgia à sua agenda.

```
obtain_better_sol(Room,Day,AgOpRoomBetter,LAgDoctorsBetter,TFinOp):-
        get_time(Ti),
        (obtain_better_sol1(Room,Day);true),
        retract(better_sol(Day,Room,AgOpRoomBetter,LAgDoctorsBetter,TFinOp)),
            write('Final Result: AgOpRoomBetter='),write(AgOpRoomBetter),nl,
            write('LAgDoctorsBetter='),write(LAgDoctorsBetter),nl,
            write('TFinOp='),write(TFinOp),nl,
        get_time(Tf),
        T is Tf-Ti,
        write('Tempo de geracao da solucao:'),write(T),nl.


obtain_better_sol1(Room,Day):-
    asserta(better_sol(Day,Room,_,_,1441)),
    findall(OpCode,surgery_id(OpCode,_),LOC),!,
    permutation(LOC,LOpCode),
    retractall(agenda_staff1(_,_,_)),
    retractall(agenda_operation_room1(_,_,_)),
    retractall(availability(_,_,_)),
    findall(_,(agenda_staff(D,Day,Agenda),assertz(agenda_staff1(D,Day,Agenda))),_),
    agenda_operation_room(Room,Day,Agenda),assert(agenda_operation_room1(Room,Day,Agenda)),
    findall(_,(agenda_staff1(D,Day,L),free_agenda0(L,LFA),adapt_timetable(D,Day,LFA,LFA2),assertz(availability(D,Day,LFA2))),_),
    availability_all_surgeries(LOpCode,Room,Day),
    agenda_operation_room1(Room,Day,AgendaR),
        update_better_sol(Day,Room,AgendaR,LOpCode),
        fail.
```

- **obtain_better_sol/5:** Este predicado principal é responsavel por chamar o obtain_better_sol1/2 para este último chamar todos os predicados chamados em cima para com isto conseguir obter uma agenda das cirurgias sem conflitos. Imprime também para comparação de desempenhos o tempo que durou a geração do resultado.

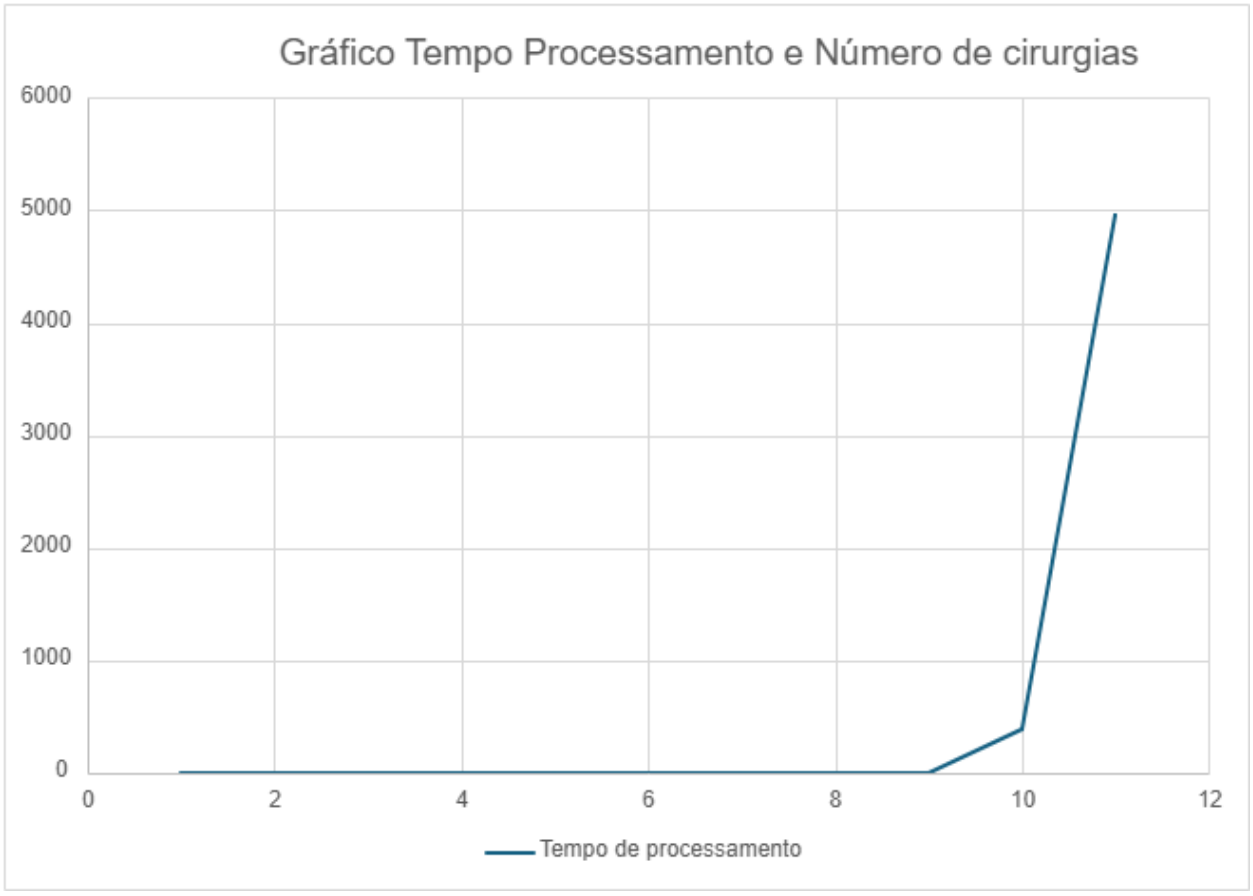
Estudo da complexidade

Nº cirurgias	Resultado	TFinOp	Tempo de geração
1	[(520, 579, so100000), (580, 639, so100001), (1000, 1059, so099999)]	639	0.01240396499633789
2	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (1000, 1059, so099999)]	729	0.020236968994140625
3	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (1000, 1059, so099999)]	804	0.044281005859375
4	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (805, 864, so100004), (1000, 1059, so099999)]	864	0.29041099548339844
5	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (1000, 1059, so099999)]	939	0.5312330722808838
6	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...)]	999	3.0457839965820312
7	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...), (..., ...)],	1149	13.66140604019165
8	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...],	1224	14.57058596611023
9	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (790, 849, so100009), (850, 909, so100001), (910, 969, so100006), (1000, ..., ...), (..., ...)|...],	1299	44.21948194503784
10	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...],	1374	388.5527129173279
11	_60	1441	4951.260046005249
12	Problema a gerar, empancou		
13	Problema a gerar, empancou		

A partir da tabela conseguimos perceber que a partir das 10 cirurgias há um tempo de processamento muito grande, pelo que consideramos que a partir daí deixa de ser viável para o utilizador.

Através da análise à tabela conseguimos perceber que este gráfico não cresce de forma linear, por isso podemos concluir que a complexidade da solução é exponencial. O processo de geração de todas as permutações de cirurgias (permutation/2) causa uma complexidade **O(n!)** no

predicado principal. As funções de interseção de agendas têm como pior caso **O(n*m**) sendo n e m os intervalos das agendas de cada um.



Podemos concluir que o tempo de processamento aumenta rapidamente com o número de cirurgias, devido à necessidade de calcular interseções entre várias agendas e combinar os horários dos médicos. Se o número de cirurgias continuar a aumentar, podemos esperar que o tempo de execução tenha uma subida ainda mais acentuada, ou seja, este método de agendamento pode não ser viável para grandes números de cirurgias.

Adaptação do método para incluir todo o staff necessários e as fases da operação

```
agenda_staff(d001,20241028,[(720,790,m01),(1080,1140,c01)]).
agenda_staff(d002,20241028,[(850,900,m02),(901,960,m02),(1380,1440,c02)]).
agenda_staff(d003,20241028,[(720,790,m01),(910,980,m02)]).
agenda_staff(n001, 20241028, []).
agenda_staff(n002, 20241028, []).
agenda_staff(d004, 20241028, []).
agenda_staff(n003, 20241028, []).
agenda_staff(t001, 20241028, []).
agenda_staff(t002, 20241028, []).



timetable(d001,20241028,(480,1200)).
timetable(d002,20241028,(500,1440)).
timetable(d003,20241028,(520,1320)).
timetable(n001, 20241028, (480, 1200)).
timetable(n002, 20241028, (480, 1200)).
timetable(d004, 20241028, (480, 1200)).
timetable(n003, 20241028, (480, 1200)).
timetable(t001, 20241028, (480, 1200)).
timetable(t002, 20241028, (480, 1200)).


% Staff responsável pela anestesia
staff(d004,doctor,anesthesia,[so2,so3,so4]).
staff(n003,nurse,anesthesia,[so2,so3,so4]).

% Staff responsável pela cirurgia
staff(d001,doctor,surgeon,[so2,so3,so4]).
staff(d002,doctor,surgeon,[so2,so3,so4]).
staff(d003,doctor,surgeon,[so2,so3,so4]).
staff(n001,nurse,surgeon,[so2,so3,so4]).
staff(n002,nurse,surgeon,[so2,so3,so4]).

% Staff responsável pela limpeza
staff(t001, technician, cleaning, [so2,so3,so4]).
staff(t002, technician, cleaning, [so2,so3,so4]).
```

Diferentemente da abordagem base neste caso em staff/4 passamos também a role quer de Nurse quer de Technician. Atribuímos também implicitamente as fases que cada um pode desempenhar. Neste caso na fase de anestesia precisa de 1 nurse e 1 doctor. Na fase de cirurgia precisa de 3 doctors e 2 nurses. Na fase da limpeza precisa de 2 technicians.

```
%surgery(SurgeryType,TAnesthesia,TSurgery,TCleaning).
surgery(so2,45,60,45).
surgery(so3,45,90,45).
surgery(so4,45,75,45).

surgery_id(so100001,so2).
surgery_id(so100002,so3).
%surgery_id(so100003,so4).
%surgery_id(so100004,so2).
%surgery_id(so100005,so4).

%surgery_id(so100006, so3).
%surgery_id(so100007, so2).
%surgery_id(so100008, so4).
%surgery_id(so100009, so3).

assignment_surgery(so100001, d001).
assignment_surgery(so100001, n001).
assignment_surgery(so100001, d004).
assignment_surgery(so100001, t001).

assignment_surgery(so100002, n001).
assignment_surgery(so100002, d002).
assignment_surgery(so100002,d004).
assignment_surgery(so100002,t002).

%assignment_surgery(so100004,d001).
%assignment_surgery(so100004,d002).
%assignment_surgery(so100005,d002).
%assignment_surgery(so100005,d003).

%assignment_surgery(so100006,d001).
%assignment_surgery(so100007,d002).
%assignment_surgery(so100008,d003).
%assignment_surgery(so100009,d003).
```

Nesta trecho de código, os predicados estão praticamente iguais, para além de inserirmos os doctors incluimos também os nurses e os technicians.

```
schedule_all_surgeries(Room,Day):-
    clean_dynamic_data,
    create_dynamic_data(Day),
    define_staff_availability,

    findall(OpCode,surgery_id(OpCode,_),LOpCode),
    availability_all_surgeries(LOpCode,Room,Day),!.


define_staff_availability() :- findall(_,(agenda_staff1(D,Date,L),free_agenda0(L,LFA),adapt_timetable(D,Date,LFA,LFA2),assertz(availability(D,Date,LFA2))),_).


create_dynamic_data(Day) :-
    findall(_,(agenda_staff(D,Day,Agenda),assertz(agenda_staff1(D,Day,Agenda))),_),
    agenda_operation_room(Or,Date,Agenda),assert(agenda_operation_room1(Or,Date,Agenda)).


clean_dynamic_data():-
    retractall(agenda_staff1(_,_,_)),
    retractall(agenda_operation_room1(_,_,_)),
    retractall(availability(_,_,_)).
```

- **schedule_all_surgeries/2:** Este predicado agenda todas as cirurgias em uma determinada sala (Room) e em um determinado dia (Day).Chama um predicado clean_dynamic_data/0 que limpa todos os dados dinâmicos previamente armazenados. Cria os novos dados de acordo com os dados em agenda_staff/3 que mostrei em cima e em agenda_operation_room o mesmo acontece. Usa findall/3 para buscar todos os códigos de cirurgia registados no sistema, definidos em surgery_id/2. Chama availability_all_surgeries/3, que verifica a disponibilidade dos staffs e preenche a agenda de acordo com a disponibilidade.

```
availability_all_surgeries([],_,_).
availability_all_surgeries([OpCode|LOpCode],Room,Day):-

    surgery_id(OpCode,OpType) , surgery(OpType, TAnesthesia, TSurgery, TCleaning),
    availability_operation(OpCode, Room, Day, Interval, LDoctorsSurgery, LStaffAnesthesia, LStaffCleaning),
    calculate_intervals(Interval, TAnesthesia, TSurgery, TCleaning, MinuteStartAnesthesia, MinuteStartSurgery, MinuteStartCleaning, MinuteEndProcess),


    retract(agenda_operation_room1(Room,Day,Agenda)),
    insert_agenda((MinuteStartAnesthesia,MinuteEndProcess,OpCode),Agenda,Agenda1),
    assertz(agenda_operation_room1(Room,Day,Agenda1)),

    insert_agenda_staff((MinuteStartSurgery,MinuteStartCleaning,OpCode),Day,LDoctorsSurgery),
    insert_agenda_staff((MinuteStartAnesthesia,MinuteStartCleaning,OpCode),Day,LStaffAnesthesia),
    insert_agenda_staff((MinuteStartCleaning, MinuteEndProcess, OpCode), Day, LStaffCleaning),

    availability_all_surgeries(LOpCode,Room,Day).

calculate_intervals((Start, End), TAnesthesia, TSurgery, TCleaning, MinuteStartAnesthesia, MinuteStartSurgery, MinuteStartCleaning, MinuteEndProcess) :-

    MinuteStartAnesthesia = Start,

    MinuteStartSurgery is MinuteStartAnesthesia + TAnesthesia,
    MinuteStartCleaning is MinuteStartSurgery + TSurgery,
    MinuteEndProcess is MinuteStartCleaning + TCleaning,

    MinuteEndProcess =< End.


availability_operation(OpCode,Room,Day,Interval,LDoctorsSurgery,LStaffAnesthesia, LStaffCleaning):-
    surgery_id(OpCode, OpType) , surgery(OpType, TAnesthesia, TSurgery, TCleaning),
    findall(Staff, (assignment_surgery(OpCode, Staff) , staff(Staff,_,surgeon,_)), LDoctorsSurgery),
    findall(Staff, (assignment_surgery(OpCode, Staff) , staff(Staff,_,anesthesia,_)), LStaffAnesthesia),
    findall(Staff, (assignment_surgery(OpCode, Staff) , staff(Staff,_,cleaning,_)), LStaffCleaning),

    intersect_all_agendas(LDoctorsSurgery, Day, LSurgery),
    intersect_all_agendas(LStaffAnesthesia, Day, LAnesthesia),
    intersect_all_agendas(LStaffCleaning, Day, LCleaning),

    agenda_operation_room1(Room, Day, LAgenda),
    free_agenda0(LAgenda, LFAgRoom),

    find_first_interval(LAnesthesia, LSurgery, LCleaning, LFAgRoom, TAnesthesia, TSurgery, TCleaning, Interval).
```

- **availability_all_surgeries/3:** Este predicado é responsável por iterar sobre todas as cirurgias e garantir que as agendas quer da sala quer do staff sejam atualizadas. Começa por definir os surgery_ids e os surgerys e depois, depois chamamos o predicado availability_operation/7 que verifica a disponibilidade do staff necessário (doctors, nurses e technicians) e das salas de operação. Retorna o intervalo de tempo disponível (Interval) e as listas de staffs disponíveis para cada função respetivamente. Chama o calculate_intervals/8 e calcula os horários de início e fim para as diferentes fases da cirurgia (anestesia, cirurgia e limpeza), com base no intervalo de tempo disponível (Interval) e nas durações de cada parte da cirurgia (TAnesthesia, TSurgery, TCleaning). Retorna os horários de início e fim das fases : MinuteStartAnesthesia,MinuteStartSurgery, MinuteStartCleaning, e MinuteEndProcess. Depois insere o novo intervalo de tempo da cirurgia na agenda da sala de operações.O insert_agenda_staff/3 atualiza as agendas do staff envolvido na cirurgia (doctors, nurses e technicians) para aquele dia. Depois o método chama-se a si mesmo recursivamente para realizar isto para todas as cirurgias. A chama do método principal é igual ao código base fornecido.

Output da solução

“Analysing for LOpCode=[so100002,so100001] now: FinTime1=911 Agenda=[(520,579,so100000),(580,760,so100002),(761,911,so100001),(1000,1059,so099999)] Best solution updated Final Result: AgOpRoomBetter=[(520,579,so100000),(580,760,so100002),(761,911,so100001),(1000,1059,so099999)] LAgDoctorsBetter=[(d001,[(720,790,m01),(806,866,so100001),(1080,1140,c01)]),(t001,[(866,911,so100001)]),(n001,[(625,715,so100002),(806,866,so100001)]),(d002,[(625,715,so100002),(850,900,m02),(901,960,m02),(1380,1440,c02)]),(d004,[(580,715,so100002),(761,866,so100001)]),(t002,[(715,760,so100002)])] TFinOp=911 Tempo de geracao da solucao:0.020900964736938477 Room = or1, Day = 20241028, AgOpRoomBetter = [(520, 579, so100000), (580, 760, so100002), (761, 911, so100001), (1000, 1059, so099999)], LAgDoctorsBetter = [(d001, [(720, 790, m01), (806, 866, so100001), (1080, 1140, c01)]), (t001, [(866, 911, so100001)]), (n001, [(625, 715, so100002), (806, 866, so100001)]), (d002, [(625, 715, so100002), (850, 900, m02), (901, ..., ...), (..., ...)]), (d004, [(580, 715, so100002), (761, ..., ...)]), (t002, [(715, ..., ...)])], TFinOp = 911.”

O algoritmo conseguiu agendar todos as cirurgias nas salas e atribui o respetivo horário aos doctors, bem como os nurses e os technicians, sem conflitos na agenda. Na variável AgOpRoomBetter temos as cirurgias que vão decorrer naquela sala e informações sobre o tempo e que surgery tem associada. LAgDoctorsBetter tem os horários especificos a cada staff (quer seja doctor,nurse ou technician) e o TFinOp tem a hora da última operação do dia.

Euristica 1

Enunciado da euristica 1 :

“A cirurgia seguinte a considerar é aquela que é realizada pelo médico que estiver disponível o mais rapidamente possível. O que é estar disponível cedo é ter tempo suficiente para concluir a cirurgia antecipadamente (por exemplo se o médico iniciar o horário às 8h00 e tiver uma reunião às 8h30 não estará disponível às 8h00 para uma consulta. ”

Na implementação da euristica 1 utilizamos como base a euristica obtain_better_sol utilizada no agendamento mais eficiente, no entanto como descrito no enunciado da euristica teremos que marcar a primeira cirurgia a partir do primeiro médico disponivel.

```
find_earliest_available_doctor(OpCode, Day, EarliestDoctor, EarliestTime) :-
    surgery_id(OpCode, OpType),
    surgery(OpType, _, TSurgery, _),
    findall(Doctor, assignment_surgery(OpCode, Doctor), Doctors),
    findall((Time, Doc), (
        member(Doc, Doctors),
        availability(Doc, Day, Availabilities),
        member((Start, End), Availabilities),
        % Compara os tempos para verificar se tem tempo suficiente para a cirurgia
        Duration is End - Start + 1,
        Duration >= TSurgery,
        Time = Start
    ), DoctorTimes),
    sort(DoctorTimes, [(EarliestTime, EarliestDoctor)|_]).
```

Para o tratamento dos horarios dos medicos para que seja possivel encontrar o primeiro médico disponivel, foi criado o predicado find_earliest_available_doctor onde se obtem o médico que está disponivel mais cedo através do timetable dele, depois verifica-se a disponibilidade através do availability.

Ao obter as “Availabilities”, estas, tal como refere no enunciado da euristica deverão ser tratadas para verificar que uma operação consiga ser executada na integra, para isso é comparado o tempo disponivel com o tempo estimado da operação, caso o tempo disponivel seja inferior ao tempo necessário para a operação esta disponibilidade será removida do availabilities.

Este processo é repetido a cada inserção de uma cirurgia. Com a ordenação por tempo mais

cedo dos tempos disponiveis de cada doutor pretende-se que na proxima iteração o algoritmo obtenha o tempo mais inicial no dia.

Caso a operação precise de mais de um doutor foi utilizado o predicado das interseções que utiliza as disponibilidades geradas previamente e faz a sua interseção até á primeira compatibilidade:

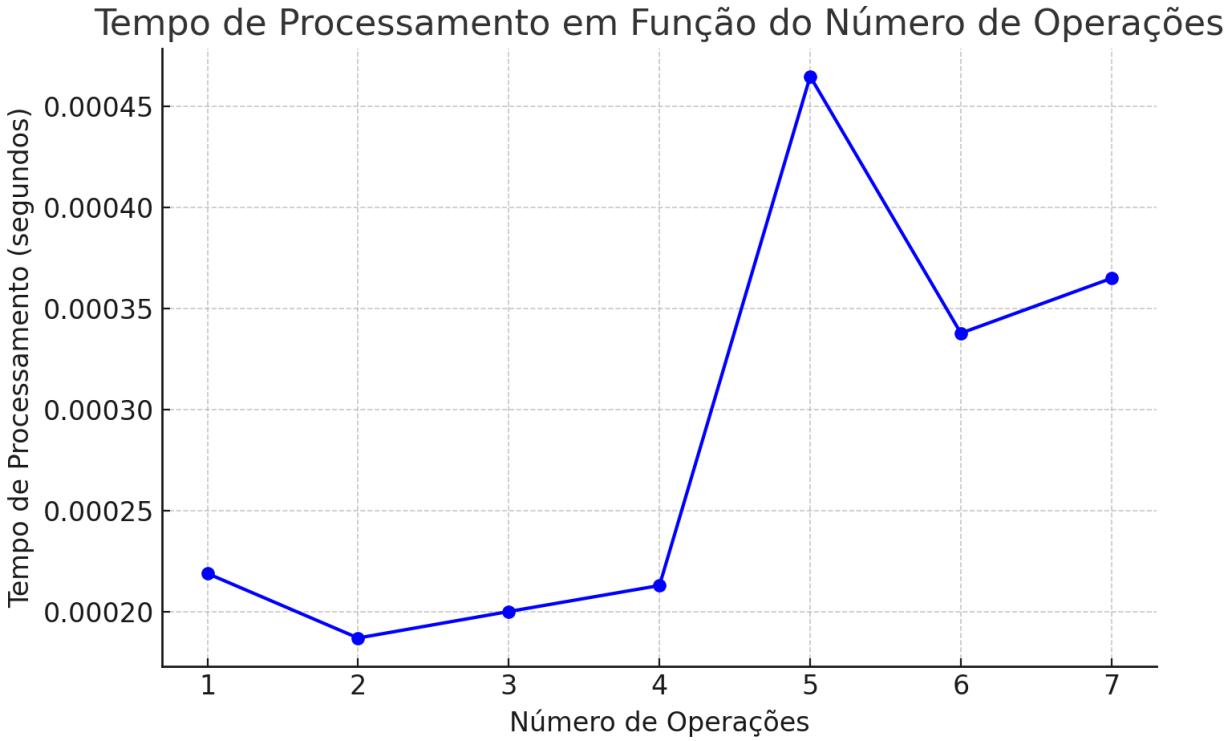
```
intersect_availability((_,_),[],[],[]).
intersect_availability((_,Fim),[(Ini1,Fim1)|LD],[],[(Ini1,Fim1)|LD]):-
    Fim<Ini1,!.
intersect_availability((Ini,Fim),[(_,Fim1)|LD],LI,LA):-
    Ini>Fim1,!,
    intersect_availability((Ini,Fim),LD,LI,LA).
intersect_availability((Ini,Fim),[(Ini1,Fim1)|LD],[(Imax,Fmin)],[(Fim,Fim1)|LD]):-
    Fim1>Fim,!,
    min_max(Ini,Ini1,_,Imax),
    min_max(Fim,Fim1,Fmin,_).
intersect_availability((Ini,Fim),[(Ini1,Fim1)|LD],[(Imax,Fmin)|LI],LA):-
    Fim>=Fim1,!,
    min_max(Ini,Ini1,_,Imax),
    min_max(Fim,Fim1,Fmin,_),
    intersect_availability((Fim1,Fim),LD,LI,LA).
```

Nº cirurgias	Resultado	TFinOp	Tempo	ResultadoE1	TFinOpE1	TempoE1
1	[(520, 579, so100000), (580, 639, so100001), (1000, 1059, so099999)]	639	0.012403964 99633789	[(520, 579, so100000), (580, 639, so100001), (1000, 1059, so099999)]	639	0.00021886 8255615234 3
2	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (1000, 1059, so099999)]	729	0.020236968 994140625	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (1000, 1059, so099999)]	729	0.00018692 0166015625
3	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (1000, 1059, so099999)]	804	0.044281005 859375	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (1000, 1059, so099999)]	865	0.00020003 3187866210 94
4	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (805, 864, so100004),	864	0.290410995 48339844	[(520, 579, so100000), (580, 639, so100001), (640, 729,	120 0	0.00021290 7791137695

	(1000, 1059, so099999)]			so100002), (791, 865, so100003), (1000, 1059, so099999), (1141, 1200, so100004)]		
5	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (1000, 1059, so099999)]	939	0.5312330722808838	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (1000, 1059, so099999), (1060, 1134, so100005), (1141, 1200, so100004)]	1200	0.000545978546142578 1
6	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...)]	999	3.0457839965820312	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (866, 925, so100006), (1000, 1059, so099999), (1060, 1134, so100005), (1141, ..., ...)]	1200	0.000337839126586914 06
7	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...), (..., ...)],	1149	13.66140604019165	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (866, 925, so100006), (1000, 1059, so099999), (1060, 1134, so100005), (1141, ..., ...), (..., ...)],	1290	0.000365018844604492 2

A partir da comparação direta entre os resultados obtidos na mesma amostra de dados entre o obtain_better_sol e a euristica 1 conseguimos perceber que os tempos de processamento são extremamente mais rápidos no entanto a eficiencia da utilização do tempo é mais reduzida. Dada esta redução de eficiência concluimos que como o aproveitamento do tempo disponivel é menor, o algoritmo não será capaz de marcar mais do que sete operações para o mesmo dia. Mesmo que esta euristica seja menos capaz na quantidade de dados tratados consideramos

que seja importante para situações em que seja priorizada a velocidade de obtenção de resposta como uma marcação de cirurgia de emergência no contexto do projeto.



Euristica 2

Enunciado da euristica 2 :

“A próxima cirurgia é para o médico que precisará estar ocupado com maior percentagem de tempo. Por exemplo, se para um determinado médico restam 2 cirurgias de 60 minutos e 1 cirurgia de 90 minutos, num total de 210 minutos, e se somarmos os tempos livres desse Médico e isso resultar em 420 minutos, isso significa que ele precisará estar ocupado 50 minutos % do tempo restante. Suponhamos que outros médicos tenham menos que esse percentual, melhor é começar com uma cirurgia envolvendo o médico mais ocupado. Observe que esses cálculos precisam ser feitos passo a passo (após o agendamento dos avisos,

suponhamos que a cirurgia de 90 minutos, a ocupação do médico será 36,4%, e então outro médico com 40% de ocupação pode ser mais crítico.”

Na Euristica 2 proposta a solução deverá ser calculada tendo em conta a percentagem de ocupação de um médico, o algoritmo deve procurar ocupar o médico mais ocupado que ainda não tenha ultrapassado a percentagem estipulada para maximo de ocupação.

```
calcular_ocupacao(Medico, Dia, Percentagem):-
    agenda_staff1(Medico, Dia, Agenda),
    calcular_tempo_ocupado(Agenda, TempoTotalOcupado),
    timetable(Medico, Dia, (Inicio, Fim)),
    TempoDisponivel is Fim - Inicio,
    Percentagem is (TempoTotalOcupado / TempoDisponivel) * 100.
```

Depois de á semelhança de na melhor solução serem encontrados os intervalos livres, o passo seguinte é identificar o médico cuja ocupação percentual será maior se a próxima cirurgia for atribuída a ele. Depois identifica o médico mais critico no calcular_percentagem_maxima.

```
calcular_percentagem_maxima(Day, LOpCode, PercentagemMaxima):-
    findall(Percentagem, (member(OpCode, LOpCode), assignment_surgery(OpCode, Medico),
     calcular_ocupacao(Medico, Day, Percentagem)), ListaPercentagem),
    max_list(ListaPercentagem, PercentagemMaxima).
```

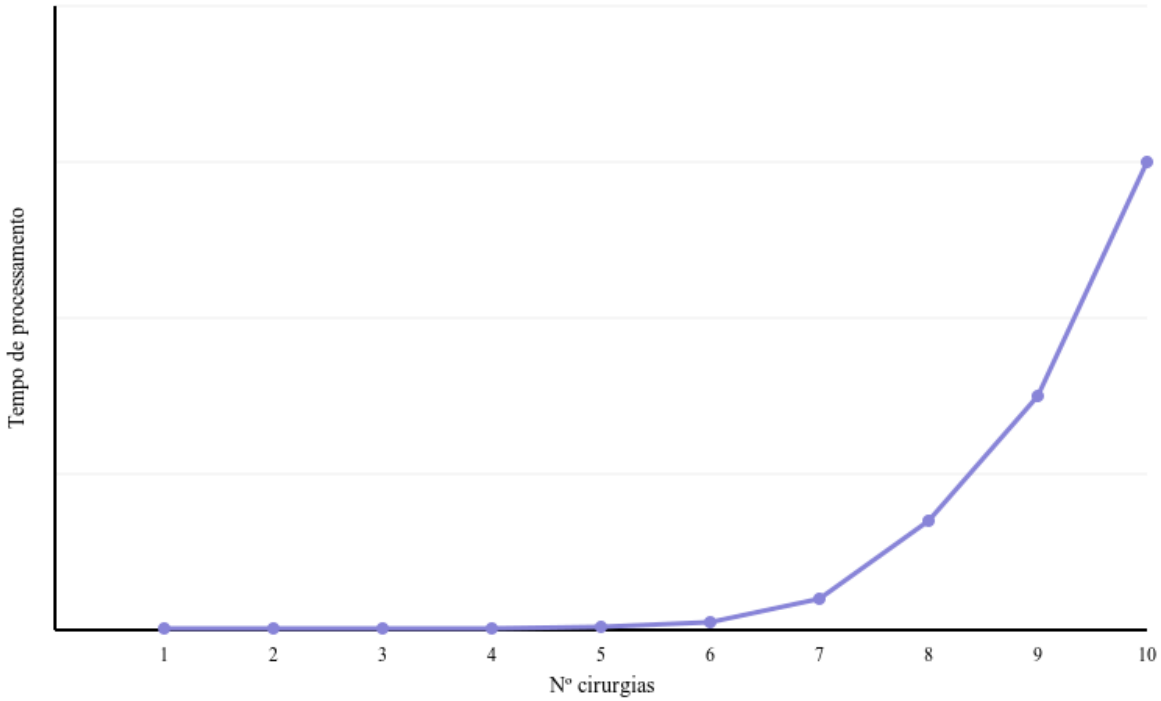
Depois é efetuado o procedimento normal explicado anteriormente para a solução mais eficiente tendo em conta que o médico priorizado pelo algoritmo será o medico mais ocupado.

Nº cirurgia	Resultado	TFimOp	Tempo	ResultadoE2	TFimOpE2	Tempo E2	%Max
1	[(520, 579, so100000), (580, 639, so100001), (1000, 1059, so099999)]	639	0.01240396499633789	[(520, 579, so100000), (580, 639, so100001), (1000, 1059, so099999)],	639	0.0037729740142822266	26.25
2	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (1000, 1059, so099999)]	729	0.020236968994140625	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (1000, 1059, so099999)],	729	0.0036411285400390625	27.4468085106383
3	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (1000, 1059, so099999)]	804	0.044281005859375	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003),	865	0.00391387939453125	27.4468085106383

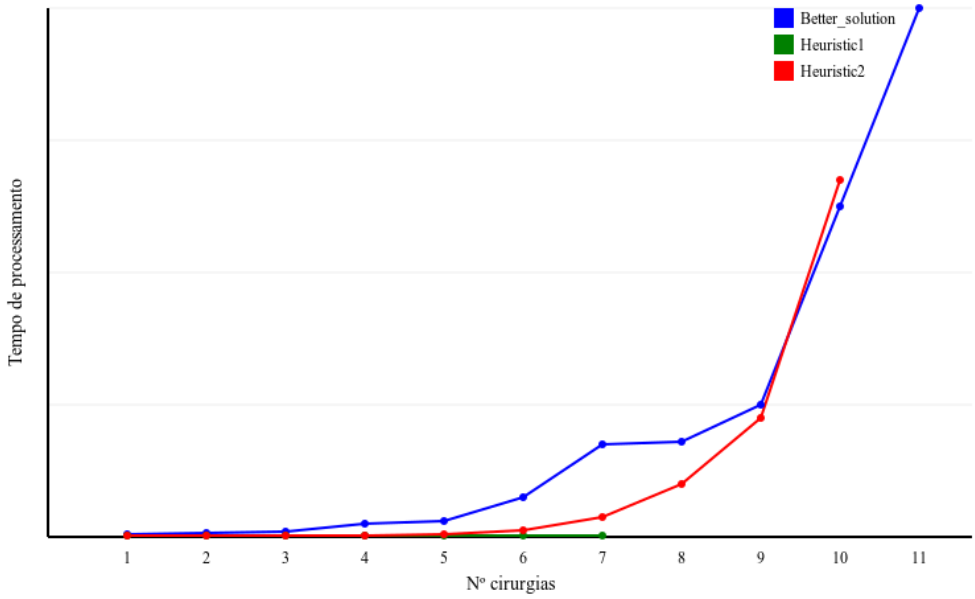
				(1000, 1059, so099999)],			
4	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (805, 864, so100004), (1000, 1059, so099999)]	864	0.29041099548339844	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (1000, 1059, so099999), (1141, 1200, so100004)],	1200	0.004362821578979492	34.4444444444444
5	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (1000, 1059, so099999)]	939	0.5312330722808838	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (1000, 1059, so099999), (1060, 1134, so100005), (1141, 1200, so100004)],	1200	0.016556978225708008	41.59574468085106
6	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...)]	999	3.0457839965820312	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (866, 925, so100006), (1000, 1059, so099999), (1060, 1134, so100005), (1141, ..., ...)],	1200	0.09235501289367676	42.638888888888886
7	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...), (..., ...)],	1149	13.66140604019165	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (866, 925, so100006), (1000, 1059, so099999), (1060, 1134, so100005), (1141, ..., ...), (..., ...)],	1290	0.6829831600189209	47.125
8	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...],			[(520, 579, so100000), (580, 639, so100001), (640, 699, so100008), (700, 789, so100002), (791, 880, so100007), (881, 940, so100006), (1000, 1059, so099999), (1060, ..., ...), (..., ...)|...],	1275	4.3432698249816895	54.500000000001
9	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (790, 849, so100009),			[(520, 579, so100000), (580, 639, so100001), (640, 699, so100008), (700,	1365	38.92283391952515	54.500000000001

	(850, 909, so100001), (910, 969, so100006), (1000, ..., ...), (..., ...)|...],			759, so100009), (791, 880, so100007), (881, 940, so100006), (1000, 1059, so099999), (1060, ..., ...), (..., ...)|...],			
10	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...],			[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...],	1374	426.158473	61.875

As conclusões obtidas através da comparação direta entre as duas euristicas revelam que a solução desenvolvida na euristica 2 apresenta uma maior eficácia do que a euristica da melhor solução com um número reduzido de operações a agendar no entanto quando o cenário pretende o agendamento de um número superior de operações esta euristica aproxima-se da melhor solução no ponto de vista de tempo de execução chegando até mesmo a verificar-se uma resposta mais lenta.
Visto que o momento em que a euristica 2 se torna menos eficaz que a melhor solução é dado com um tempo de execução não viavel ao utilizador nas duas euristicas concluimos que a euristica 2 realizada será a melhor solução a adotar pois apresenta a melhor relação eficácia e eficiência.


Tempo de processamento
1
2
3
4
5
6
7
8
9
10
Nº cirurgias

Conclusão



Através do gráfico conseguimos comparar 3 abordagens diferentes para gerar horários para staffs e para as salas de cirurgia.

- **Better_solution (linha azul):** Apresenta a melhor solução a nível de eficiência no uso de tempo no agendamento, mas o tempo de processamento aumenta significativamente à medida que o número de cirurgias cresce. Torna-se menos útil para cenários com muitas cirurgias, pois o tempo de execução será demasiado elevado.

- **Heuristic1 (linha verde)**: Mantém o tempo de processamento constantemente baixo. No entanto, quando o número de cirurgias aumenta e uma cirurgia requer mais do que um médico, como a eficiência na gestão do tempo é muito reduzida,o algoritmo tem dificuldade em fazer as interseções de diferentes agendas. Pode ser a melhor escolha para sistemas que priorizam tempo de resposta como uma marcação de cirurgia de emergência, mas pode comprometer a qualidade da solução em comparação com a abordagem "Better_solution".

- **Heuristic2 (linha vermelha):** Este algoritmo, oferece um equilíbrio entre tempo de processamento e qualidade da solução, apresentando um tempo de execução aceitável até cerca de 10 cirurgias. Para mais de 10 cirurgias não foi possível obter mais respostas porque o algoritmo não conseguia conciliar as agendas.